Proceedings of the 6th International Conference on Gas Hydrates (ICGH 2008),
Vancouver, British Columbia, CANADA, July 6-10, 2008.

A DOMAIN DECOMPOSITION APPROACH FOR LARGE-SCALE SIMULATIONS OF FLOW PROCESSES IN HYDRATE-BEARING GEOLOGIC MEDIA

Keni Zhang*, George J. Moridis, Yu-Shu Wu, and Karsten Pruess
Earth Sciences Division
Lawrence Berkeley National Laboratory
1 Cyclotron RD, Berkeley, CA 94720, USA

ABSTRACT

Simulation of the system behavior of hydrate-bearing geologic media involves solving fully coupled mass- and heat-balance equations. In this study, we develop a domain decomposition approach for large-scale gas hydrate simulations with coarse-granularity parallel computation. This approach partitions a simulation domain into small subdomains. The full model domain, consisting of discrete subdomains, is still simulated simultaneously by using multiple processes/processors. Each processor is dedicated to following tasks of the partitioned subdomain: updating thermophysical properties, assembling mass- and energy-balance equations, solving linear equation systems, and performing various other local computations. The linearized equation systems are solved in parallel with a parallel linear solver, using an efficient interprocess communication scheme. This new domain decomposition approach has been implemented into the TOUGH+HYDRATE code and has demonstrated excellent speedup and good scalability. In this paper, we will demonstrate applications for the new approach in simulating field-scale models for gas production from gas-hydrate deposits.

Keywords: gas hydrate, reservoir simulation, domain decomposition, parallel computing

INTRODUCTION

Mathematical models (or numerical models) are effective tools for understanding the behavior of gas hydrate systems in nature or in the laboratory. Numerical modeling can specifically play an important role in evaluating schemes for gas production from different types of hydrate accumulations. Simulation of gas hydrate reservoir production involves solving a set of highly nonlinear, coupled fluid-, heat-, and mass-transport equations, combined with the potential for formation and/or disappearance of multiple solid phases in a system typical of common natural hydrate deposits (either in permafrost or in deep ocean sediments) within complex geological media—at any scale (from laboratory to reservoir). Also, the physical and chemical properties of the geologic media containing gas hydrate are highly dependent on the amount of gas hydrate present in the system at any given time. In most cases, hydrate dissociation for gas production shows very strong nonlinearity. In general, modeling of such a complex system of gas hydrate dissociation and migration processes requires fine spatial and temporal discretization, and therefore presents a significant computational challenge.

There are a limited number of numerical codes for simulating gas production from hydrate systems. At present, relatively mature simulators may include the MH-21 hydrate reservoir simulator, STOMP-HYD, CMG STARS, and TOUGH+HYDRATE. Numerous groups have developed their own simulators for use on specific examples and classes of laboratory through field hydrate-related experiments and characterizations.

One of the most widely used gas hydrate simulators is TOUGH+HYDRATE [1], a module of the TOUGH+ code. TOUGH+ is the successor to the TOUGH2 suite of codes [2]. *Moridis et al.* [3] developed EOSHYDR, a TOUGH2 module for the simulation of dissociating simple methane hydrates under equilibrium conditions in both permafrost and marine accumulations. In 2003, EOSHYDR was enhanced to EOSHYDR2 for the simulation of binary hydrates reacting under both equilibrium and kinetic conditions. EOSHYDR2 was further improved to the current version of TOUGH+HYDRATE [1].

The TOUGH+ HYDRATE simulator has been successfully used for the simulation of gas production from hydrates under a variety of geologic and thermodynamic conditions, and involving various production strategies [4, 5, 6, 7]. The current version of the code can simulate both equilibrium and kinetic models of hydrate formation and dissociation. The model accounts for heat and up to four mass components (water, CH4, hydrate, and water-soluble inhibitors such as salts or alcohols). These are partitioned among four possible phases (gas phase, liquid phase, ice phase, and hydrate phase). Hydrate dissociation or formation, phase changes, and the corresponding thermal effects are fully described, as are the effects of inhibitors. TOUGH+ HYDRATE can describe all possible hydrate dissociation mechanisms—depressurization, thermal stimulation, salting-out effects, and inhibitor-induced effects. Because of the complexity of subsurface flow processes and phase changes, most simulations for gas production from hydrate accumulations are limited to systems of up to several thousand gridblocks. For field scale applications or refined experimental models, however, hundred thousands and even millions of gridblocks may be needed to represent geologic heterogeneities, multiphase flow, phase behavior, and, multicomponent flow structures on different scales.

In this study, we discuss the development of a parallel numerical modeling approach for large-scale simulations of flow processes in hydrate-bearing geologic media. This new development is based on the current version of TOUGH+HYDRATE code. Parallelization of the TOUGH+ framework uses a similar approach to that used in the development of TOUGH2-MP, the parallel version of TOUGH2 [8,9]. A domain decomposition approach and MPI (Message Passing Interface) are used for the parallel implementation. In this approach, the simulation domain, defined by an unstructured grid, is partitioned into a number of subdomains using a partitioning algorithm from the METIS software package [10]. Each subdomain is handled by one processor for updating thermophysical properties, assembling mass- and energy-balance equations, solving linear equation systems, and performing other local computations. Local linear-equation systems are solved in parallel by multiple processes with the Aztec linear solver package [11].

The efficiency and scalability of the developed parallel scheme are demonstrated by examples of various applications. These examples indicate that the parallel simulator enables much larger problems to be solved, as well as significantly improving modeling capability in terms of simulation time and problem size. The code also demonstrates excellent scalability. In this paper, we present a high-resolution simulation of gas production from a Class 3 hydrate accumulation.

MATHEMATICAL MODELS

It is well known that under suitable conditions of low temperature and high pressure, a hydrocarbon gas will react with water to form hydrates following the reaction:

$$X + N_H H_2O = X \bullet N_H H_2O \qquad (1)$$

where N_H is the hydration number. For CH_4-hydrate, X represents methane. The hydrate number varies between 5.75 (for complete hydration) and 7.2 [12] for CH_4-hydrate. According to the thermodynamic condition of a system, the amount of gas hydrate created or gas released can be determined from the reaction represented by Equation (1). Natural hydrates may consist of multiple gases. TOUGH+HYDRATE simulates the formation of a composite hydrate according to [1]:

$$\begin{aligned} &\chi_m[CH_4 + N_m H_2O] + \chi_G[G + N_G H_2O] = \\ &\chi_m CH_4 + \chi_G G + (\chi_m N_m + \chi_G N_G) H_2O \end{aligned} \qquad (2)$$

where G is the second hydrate-forming gas, N is the hydration number, χ is the mole fraction in the binary hydrate, and the subscripts m and G denote methane and the second gas, respectively. Obviously, $\chi m + \chi G = 1$. The gas G may be CO_2, H_2S, N_2, or another gaseous alkane.

A hydrate-bearing geologic system can be fully described by mass-balance equations and an energy-balance equation. The mass components are partitioned among different phases. The parallel simulation solves the same equation systems as those solved by the original TOUGH+HYDRATE. The basic mass- and energy-balance equations solved can be written in the general integrated form [2]:

$$\frac{d}{dt}\int_{V_n} \mathbf{M}^\kappa dV_n = \int_{\Gamma_n} \mathbf{F}^\kappa \bullet \mathbf{n} d\Gamma_n + \int_{V_n} q^\kappa dV_n \qquad (3)$$

The integration is over an arbitrary subdomain $\boldsymbol{V_n}$ of the flow system under study, which is bounded by the closed surface Γ_n. The quantity $\boldsymbol{M}$ appearing in the accumulation term (left-hand side) represents mass or energy per volume, with k= 1, ..., NK the mass components, and k= NK + 1 the heat "component." $\boldsymbol{F}$ denotes mass or heat flux, q denotes sinks and sources, and **n** is a normal vector on surface element $d\Gamma_n$, pointing inward into **V**n.

In hydrate simulation, the components of hydrate, water, CH_4, water-soluble inhibitor (salt or organic substance) and the flow of heat are considered. These components can assume any of the following four phases: solid hydrate, aqueous, gaseous, and solid ice. In Equation (3), the mass accumulation term is evaluated by summing over all fluid phases for each mass component. The advective flux for the mass flux term in the equation is computed with a multiphase version of Darcy's law; the diffusive flux is computed by Fick's Law. Readers may refer to [1] for detailed discussions of the evaluation of mass and flux terms in the code.

Time and space discretization for Equation (3) results in a set of coupled nonlinear equations, which can be written in a residual form [2]:

$$R_n^\kappa(x^{t+1}) = M_n^\kappa(x^{t+1}) - M_n^\kappa(x^t) - \frac{\Delta t}{V_n}\{\sum_m A_{nm} F_{nm}^\kappa(x^{t+1}) + V_n q_n^{\kappa,t+1}\} = 0 \qquad (4)$$

where the vector x^t consists of primary variables at time t, R_n^κ is the residual of component κ for block n, M denotes mass or thermal energy per unit volume for a component, V_n is the volume of the block n, q denotes sinks and sources of mass or energy, Δt denotes he current time step size, $t + 1$ denotes the current time, A_{nm} is the interface area between blocks n and m, and F_{nm} is the flow between them. Equation (4) is solved by Newton/Raphson iteration, leading to

$$-\sum_i \left.\frac{\partial R_n^{\kappa,t+1}}{\partial x_i}\right|_p (x_{i,p+1} - x_{i,p}) = R_n^{\kappa,t+1}(x_{i,p}) \qquad (5)$$

where $x_{i,p}$ represents the value of i^{th} primary variable at the p^{th} iteration step.

To solve Equation (5), a set of primary variables for each gridblock/elements will be obtained at each iteration/time step. The thermophysical properties, named secondary variables, are calculated from the latest updated primary variables obtained by solving Equation (5) at the current iteration step. The secondary variables are then used to assemble the mass- and energy-balance equations for all volume elements for next iteration/time-step calculation.

PARALLELIZATION SCHEMES

Domain decomposition methods (DDM) are used as a divide-and-conquer strategy for solving large or time-consuming problems. The idea behind this approach is to divide the computational domain into a series of subdomains. Through the local solutions on the subdomains, a global solution is formed. Solutions for subdomains can be sought simultaneously. Therefore, this approach is suitable for parallel computations as long as the computational work can be evenly distributed.

We use DDM for the TOUGH+HYDRATE parallelization. The parallel implementation for solving Equation (3) first partitions the simulation domain. After domain decomposition, computations (including assembling the Jacobian

matrix, solving linear equations, and updating thermophysical properties) are done at local subdomain level by different processors. All processors involved in solving Equation (3) will solve similar equation systems for different subdomains. This technique guarantees that large sections of sequential TOUGH+HYDRATE code are reused. Each processor solves only part of the entire modeling domain; however, the best convergence performance of Newton iteration can only be achieved when the local equation systems are solved simultaneously as part of the whole system. Doing so therefore requires a parallel linear solver that facilitates intensive and expensive communication among neighboring processors.

In addition, during the simulation, communication between processors is needed for updating border thermophysical properties, collecting extreme values, conditioning across the whole simulation domain, and input/output. Extensive global communication may lead to poor scalability of the parallel code. Therefore, efficiency of the parallel code depends on code efficiency in both computation and communication. Equation (3) shows that only the flow term needs information from neighboring gridblocks for mass and energy conservation computation. The mass accumulation terms and sink or source terms are element-by-element and independent of neighboring domains. Computations related to the two terms do not require communication between neighboring processors. Because we use similar computation schemes in the parallel code and the serial code, as long as the computational work is evenly distributed, parallel speedup of the code is predominantly determined by communication efficiency.

Domain partitioning

Developing an efficient and effective method for partitioning unstructured grid domains is a first and critical step for a successful parallel scheme. To obtain optimal performance, the partitioning algorithm should ideally take the following five issues into account: (1) balancing computational load; (2) minimizing the average volume of interdomain communication; (3) balancing communication volume load; (4) minimizing the average number of neighboring processors; and (5) balancing the number of neighboring processors across all domains. To find an optimal trade-off among these five issues, computer system characteristics, such as floating-point performance, bandwidth and the latency of the communication subsystem, all must be taken into account. Commonly used algorithms and software for partitioning large grids do not generally take all these five issues into account. The typical current practice finds a trade-off between computational load balancing and maintaining low total communications volume, even though the result may not be theoretically optimal.

In a TOUGH+HYDRATE simulation, a model domain is represented by a set of three-dimensional gridblocks (or elements) and the interfaces between any two gridblocks are represented by connections. The entire connection system of gridblocks is treated as an unstructured grid. We utilize one of the three partitioning algorithms provided by the METIS package [10] for partitioning the grid domain. The three algorithms have different objectives: for minimizing the number of edges that straddle different partitions and for minimizing the total communication volume. METIS provides an effective approach for unstructured grid partitioning for large-scale simulations.

Assembly of Jacobian matrix

In the TOUGH+HYDRATE formulation, discretization of mass and energy conservation equations in space and time using the integral finite-difference (IFD) method leads to a set of strongly coupled nonlinear algebraic equations, solved by the Newtonian method. The resulting system of linear equations is then solved using a linear solver.

The Jacobian matrix needs to be recalculated at each Newton iteration step, and thus the computational effort may be considerable for a large-scale simulation. In the parallel code, the assembly of the linear equation system is shared by all the processors, with each processor responsible for computing the rows of the Jacobian matrix that correspond to the block assigned to the processor. Computation of the elements in the Jacobian matrix is performed in two parts.

- Computations related to individual blocks (mass accumulation, and source/sink terms). Such calculations are carried out using the information stored on the current

processor; no communications with other processors are needed.

- All computations related to the connections or flow terms. Calculation of flow terms for gridblocks located at the border of a subdomain requires information from its neighboring subdomain, which in turn requires communication with neighboring processors. Before performing these computations, an exchange of relevant primary variables is necessary.

Solving of linear equations

The linearized equation system arising after each Newton step is solved using a parallel iterative linear solver from the Aztec package [11]. This package includes several different solvers and preconditioners, including conjugate gradient, restarted generalized minimal residual, conjugate gradient squared, transposed-free quasi-minimal residual, and bi-conjugate gradient with stabilization methods. The work of solving the global linearized equation is shared by all processors, with each processor responsible for performing calculations within its own portion of the partitioned subdomain.

During a simulation, time steps are automatically adjusted (increased or reduced) depending on the convergence rate. In the parallel TOUGH+HYDRATE code, the time-step size is calculated by the master processor, after necessary data are collected from all other processors (since the convergence rates may be different in different processors). Only when all processors reach stopping criteria will the algorithm proceed to the next time step. Periodic output of simulation results is handled by the master processor, which gathers data from each domain and compiles the system-wide output files.

Updating thermophysical properties

The thermophysical properties of the system (secondary variables) that are needed for assembling the governing mass- and energy-balance equations are calculated at the end of each Newton iteration step, based on the updated set of primary parameters. At the same time, phase conditions are evaluated for all gridblocks, the appearance or disappearance of phases are recognized, and primary variables are switched and properly re-initialized in response to any change of phase. All these tasks must be done gridblock by gridblock across the whole simulation domain. The computational work for these tasks is readily parallelized, as each processor handles its corresponding subdomain. Some overlapping computations are needed for gridblocks that lie at subdomain borders, although this avoids additional communication for secondary variables.

Communication among processors

The exchange of data among processors working on connected gridblocks that span different domains is an essential component of the parallel algorithm. Moreover, global communication is also required to compute norms of vectors, contributed by all processors, for checking the convergence. In addition to the communication occurring inside the linear solver routine to solve the linear equation system, communication among neighboring processors is necessary to update primary variables. A communications subroutine is used to manage data exchange among processors. When this subroutine is called by a given processor, the code performs an exchange of vector elements of the gridblocks (at that processor's subdomain border) with the connected gridblocks within the adjacent domains. During time stepping or Newton iteration, exchange of external variables is required for the vectors containing the primary variables. More discussion on the prototype scheme used for data exchange is given in Elmroth et al. [13]. In addition, we have further improved the schemes by introducing nonblocking communication to the Aztec package and Newton iterations [14].

All data input and output are carried out through the master processor. For extremely large-scale problems, outputs may be performed by all processors involved in the computation, with each processor outputting its own portion of the simulation results. This approach can avoid excessive communication during data output intervals. For the time series outputs, a time/memory trade-off scheme is used to avoid heavy communication. This scheme stores time-series outputs in memory for a certain number of time steps (for example, 1,000 time steps); then the outputs are all sent simultaneously to the master processor and written out. This method is

extremely efficient for high-latency computer systems.

In this parallel approach, the most time-consuming computation work (assembling the Jacobian matrix, updating thermophysical parameters, solving the linear equation systems, and computing chemical reactions) is distributed among all processors. The memory requirements (relative to the size of the grid and the number of coupled equations) are also distributed between multiple processors. Distributing both computing and memory requirements is essential for solving large-scale field problems and obtaining better parallel simulation performance.

APPLICATION EXAMPLES

Performance of the parallel simulator is investigated through two examples. The first example is adopted from an example in [1]. We use this first example to validate the parallel code against the existing serial TOUGH+HYDRATE. Parallel performance of the simulator is then demonstrated through comparison with performance of the serial version of the code. The second example demonstrates the application of the parallel code to a large-scale, high-resolution simulation of gas production from a Class 3 hydrate accumulation.

Equilibrium hydrate dissociation in a 2D system

The first example demonstrates a two-dimensional simulation for gas production from an areal hydrate-bearing formation. The 2D domain consists of a square system with a side of 50 m and a formation thickness of 10 m. The simulation domain is considered to be one quarter of a larger square system with a production well located at its center. Because of symmetry, it is sufficient to simulate one quarter of the large square domain. In the simulation domain, the well is located at one domain corner. The simulation domain is discretized into 2,500 gridblocks with 4,900 connections. Each gridblock has a size of 1×1 m. The four sides, top, and bottom of the domain are all treated as no flow boundaries. By ignoring heat contribution from the boundaries, this simulation provides the worst-case scenario of gas production from such a hydrate accumulation.

The porous formation has a porosity $\varphi = 0.3$ and a permeability of $2.96\times10^{-13}\text{m}^2$. In the presence of the ice and hydrate solid phases, the critical mobile porosity (below which the porous medium becomes impermeable) is 0.05, and the porosity reduction exponent is 3. A pore compressibility of 10^{-8} 1/Pa and a thermal conductivity of 3.1 W/m/^{o}K are assigned to the porous medium. In hydrate simulation, evolution of solid phases of lower density (such as ice and hydrate) can lead to extraordinarily high pressures, because the aqueous phase disappears if pore compressibility is small.

The hydrate properties for this simulation, including thermal conductivity, specific heat and density of the CH4 hydrate, are from Sloan [12], and are constants, because no information is available on their dependence on temperature and/or pressure. The simulator assumes that the constant input density of the CH4 hydrate is that at the quadruple point, and the hydrate density in the simulations is internally adjusted by assuming that its compressibility and thermal expansivity are the same as those of ice. No inhibitor is used, and only equilibrium dissociation is considered in this example.

In this deposit, water, gas, and hydrate are initially at equilibrium, and the pressure is the hydration pressure corresponding to $T = 12.5^o$C. The initial gas-, aqueous-, and hydrate-phase saturations are $SG = 0.1$, $SA = 0.3$ and $SH = 0.6$, respectively. Fluids are withdrawn at a mass flow rate of $Q = 0.1$ kg/s through the production well, and are distributed in the production stream according to their mobilities. The fluid withdrawal causes a pressure decline that leads to the depressurization-induced release of CH4. The production flow rate remains constant, and is certain to lead to temperature decline and ice appearance because of the endothermic nature of dissociation.

Simulations were conducted on a 10-node Linux cluster, with each node equipped with two INTEL Xeon 3.60 GHz CPUs. The simulation was run for 100 days. Both the serial and parallel simulators are used for the simulation. Figure 1 shows the cumulative volume of CH_4 released from dissociation in the domain in the first 100 days of production. The results obtained with different number of processors are almost identical. During the simulation period, the increase of gas production rate was observed with a rate around 2,000 ST m^3 per day at the beginning to 6,000

STm^3 per day at the end. This increasing rate may be caused by the fact that at the beginning, only "free" gas can be produced, and later both free gas and gas from surrounding hydrate dissociation are released. The gas release rate is expected to change at later times, when exhaustion of the free gas and hydrate resources in the reservoir will inevitably lead to a worse production performance. Figure 2 shows simulated distribution of gas-hydrate saturation at 100 days by parallel simulator. By comparing to the corresponding result from the serial simulator, we observe that the results are not obviously different. The figure indicates severe hydrate dissociation happened within a range of 16 m. Outside of this range, we find a hydrate saturation of 0.4–0.45, which is around 2/3 of the initial gas hydrate saturation.

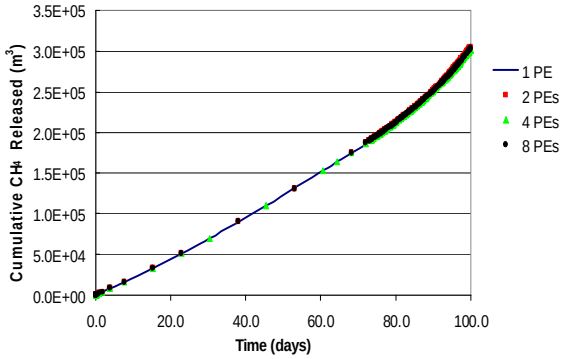

Figure 1. Simulation results of cumulative volume of CH_4 released from dissociation using different numbers of processors. The 1-processor case was run with the serial code; other cases were run with the parallel simulator.

Number of processor	1*	2	4	8	14
Total time for the simulation (s)	148	104	36	24	19
Total time steps	195	232	156	194	225

*run by the serial code

Table 1. Total time and Newton steps for the simulation with different number of processors

Table 1 shows performance of the parallel code compared to the serial code for solving this problem. The parallel simulator performs well for even such a small problem. Total execution time drops from 148 seconds using 1 processor to 19 seconds using 14 processors. Time reduction is thus quite significant, considering the small size of the problem.

The simulator uses an automatic adjustment of time-step size based on the convergence behavior of Newton iterations: time-step size will increase for fast convergence and decrease for slow convergence. Note (Table 1) that the total time steps are different with different numbers of processors. In general, when a subdomain is too small (i.e., too few gridblocks), more time steps are needed for the same length simulation. The total number of time steps for the simulation may also be influenced by the domain decomposition method. We notice that the simulation needs 232 time steps when using 2 processors with a default K-way domain partitioning method [10]. If the VK-way or a recursive method is used, total time steps and execution time will be different. The VK-way needs 180 time steps over 81 seconds, and the recursive method needs 216 time steps over 98 seconds to finish the run. Total time steps needed for the simulation may also be influenced by selection of linear solver, preconditioner, or other options for solving the linear equations. This indicates that a careful selection of different options for domain decomposition or solving linear equations may improve the code performance.

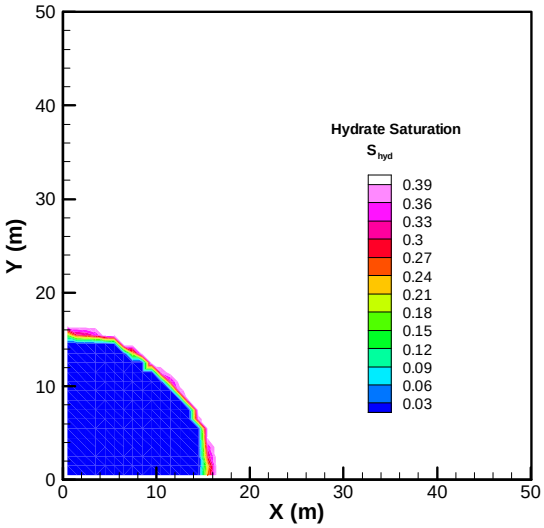

Figure 2. Hydrate saturation distribution in the reservoir at time =100 days

Gas Production from a Class 3 Hydrate Accumulation

The geologic system of this example is based on the Tigershark deposit [15], located in the

Alaminos Canyon Block 818 of the Gulf of Mexico. At this location, there is an 18.25 m thick sandy HBL (hydrate-bearing layer) with a porosity of about 0.30 and Darcy-range intrinsic permeability, at a location where the water depth is about 2,750 m. Initial estimates of gas hydrate saturation derived from geophysical methods indicated $0.6 < S_{H0} < 0.8$. Preliminary calculations indicated that the base of the gas hydrate stability zone at this location is located at or slightly below the base of the HBL.

The properties and conditions pertaining to the reference geologic system can be found in Moridis et al. [16]. The system was conceptualized as a Class 3 deposit, for which accumulations are composed of a single zone, the HBL, and are characterized by the absence of an underlying zone of mobile fluids [17]. The system has a 30 m thick impermeable upper boundary, an 18.25 m thick HBL, and a 45 m thick impermeable lower boundary (Figure 3). The horizontal well system used in this study has an area of 1000 m×1000 m. Because of symmetry involving adjacent wells, no-flow (of fluids and heat) boundaries are located at x = 0 and x = 500 m, thus necessitating simulation of only half of the domain. In this case, the only horizontal well we considered is at Location W_T (Figure 3), which has a radius r_w = 0.1 m.

The 2D domain in Figure 3 was discretized into 200×107 = 21,400 gridblocks, of which 21,200 were active (the remaining being boundary cells). The vicinity of the wellbore uses a very fine discretization along the x direction with Δx =< 7 m. The uppermost and lowermost layers correspond to constant-temperature no-flow boundaries, whereas the layers corresponding to the top and bottom confining layers are impermeable, but allow heat exchange between the deposit and its surroundings. The HBL was subdivided into segments of Δz = 0.25 m each along the z-direction. For the initial conditions, the pressures in the oceanic subsurface were assumed to follow a hydrostatic distribution. The initial temperature distribution was estimated by using the known temperature at the mud line (reported as 5°C) and the local geothermal gradient (dT/dz = 0.03464 K/m). Gas is produced by placing a horizontal well at the top of the HBL (location W_T) in which a constant pressure P_W = 3 MPa was maintained until the exhaustion of the hydrate. Detailed discussion of the problem setup was presented in Moridis et al. [16].

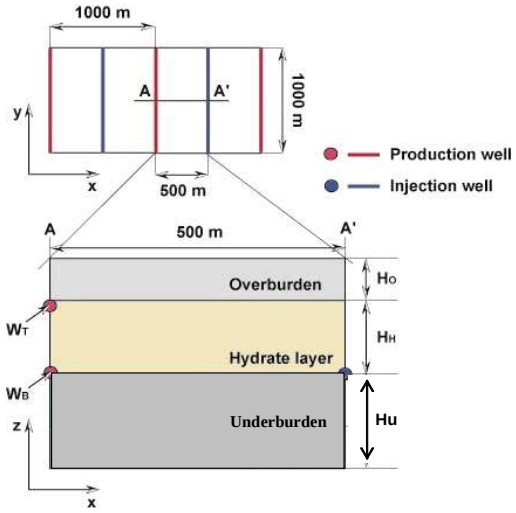

Figure 3. Geometry and configuration of Example 2. (Modified from [16])

To investigate the parallel simulator's performance, both parallel and serial version of TOUGH+HYDRATE were run on a cluster equipped with Intel Woodcrest 2.66 GHz dual-core processors. Figure 4 shows speedup of the parallel code for the first 300-day simulation. The speedup was calculated by comparing to the serial code performance, which was assumed to be 1.0. Total time needed for the simulation using one processor is 29,100 seconds with the serial code, using 32 processors with the parallel code, reduced the time to 1,366 seconds. In general, the parallel performance is good. Super liner speedup phenomenon can be observed for doubling processor number from 8 to 16 and from 16 to 32, by reducing simulation run time by more than half. Total time steps for running the 300-day simulation with different numbers of processors are in a range of less than 5% difference (around 860 time steps). All simulation runs use the same domain partitioning algorithm and linear solver options. The scalability of the code is good for the current problem up to 32 processors (without further tests with more processors, due to the limitation on available processors in that cluster). We have tested the code for another example with one-half million gridblocks; speedup can be seen on as many as several hundred processors.

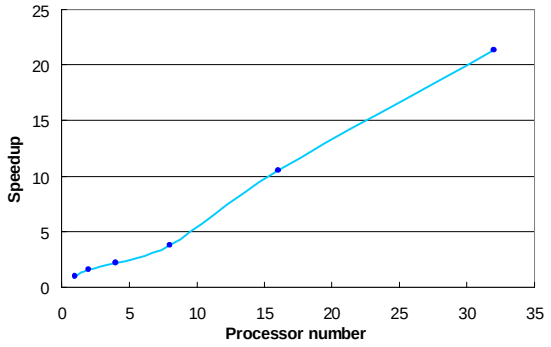

Figure 4. Speedup of the parallel simulation compared to serial code performance

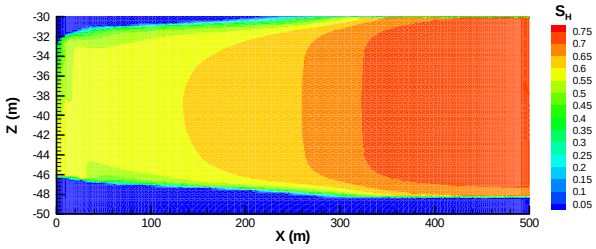

Figure 5. Simulated hydrate saturations at time=300 days

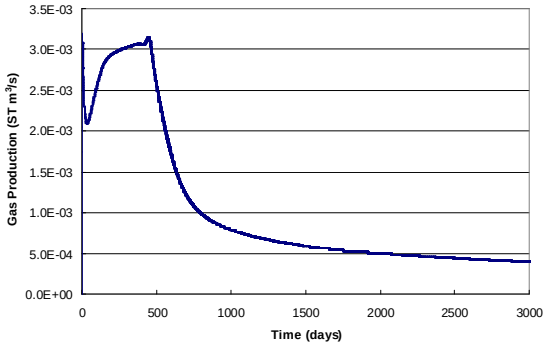

Figure 6. Evaluation of gas production rate from the HBL through the constant pressure well

Figures 5 shows simulated hydrate saturation S_H at time=300 days. The figure indicates smooth saturation gradients and the evolution of the two horizontal interfaces along the top and bottom boundary of the HBL that are typical of depressurization. Figure 6, showing evaluations of gas production rate from the well (W_T) with constant pressure, indicates a relatively high initial production increasing to maximum rate at about t = 450 days, from which it then rapidly declines. These results are identical to results obtained from the serial code simulation.

CONCLUSIONS

A domain decomposition parallel simulation approach for large-scale modeling studies of flow processes in hydrate-bearing geologic media has been developed and implemented into the current version of the TOUGH+HYDRATE simulator. The developed parallel simulator is a multidimensional, fully implicit model that solves large, sparse linear systems arising from discretization of the partial differential equations for mass and energy balance in the porous and fractured media of hydrate-bearing formations. The simulator retains all the process-modeling capabilities, input/output setup, error handling, and other features of the original TOUGH+HYDRATE, guaranteeing robustness of the parallel simulator.

The developed parallel simulator is shown to be computationally efficient. The efficiency and scalability of the code were demonstrated by field-scale examples, which show that gas production from different hydrate-bearing formation with different production schemes can be effectively simulated using the parallel simulator. It is possible to save more than 90% simulation time for a middle size model by running the code on a typical cluster with a couple dozen processors. The new simulator provides a powerful tool with which to tackle larger-scale, more complex problems than can be solved currently by sequential codes. The parallel simulator will enhance modeling capacity in terms of model size and simulation time by 1–3 orders of magnitude.

ACKNOWLEDGMENTS

The authors would like to thank Matthew Reagan and Dan Hawkes for their review of this paper. This work was supported by the Assistant Secretary for Fossil Energy, Office of Natural Gas and Petroleum Technology, through the National Energy Technology Laboratory, under the U.S. Department of Energy, Contract No. DE-AC02-05CH11231.

REFERENCES

[1] Moridis, G.J., Kowalsky, M.B., Pruess, K., *TOUGH-Fx/HYDRATE v1.0 User's Manual: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media*, Report LBNL-

3185, Lawrence Berkeley National Laboratory, Berkeley, CA, 2005.
[2] Pruess, K, Oldenburg, C. and Moridis, G., *TOUGH2 User's Guide, V2.*, Lawrence Berkeley National Laboratory Report LBNL-43134, Berkeley, CA, 1999.
[3] Moridis, G.J., Apps, J., Pruess, K., and Myer L., *EOSHYDR: A TOUGH2 Module for CH4-Hydrate Release and Flow In the Subsurface*, Report LBNL-42386, Lawrence Berkeley Laboratory, Berkeley, CA, 1998.
[4] Moridis, G.J., *Numerical studies of gas production from methane hydrates*; Society of Petroleum Engineers Journal, 2003, 32 (8), 359–370.
[5] Moridis, G.J. and Collett, T., *Gas Production from Class 1 Hydrate Accumulations*, in Recent Advances in the Study of Gas Hydrates, C. Taylor and J. Qwan, Editors, Kluwer Academic/Plenum Publishers (Section I, Chapter 6, pp. 75–88), 2004.
[6] Moridis, G.J., Reagan, M.T., Kim, S.J., Seol, Y., and Zhang, K., *Evaluation of the gas production potential of marine hydrate deposits in the Ulleung Basin of the Korean East Sea*, SPE 118859, 2007 SPE Asia Pacific Oil & Gas Conference and Exhibition held in Jakarta, Indonesia, 30 October–1 November 2007.
[7] Reagan, M.T., Moridis, G.J., Zhang, K., *Sensitivity analysis of gas production from Class 2 and Class 3 hydrate deposits*, OTC 19554, 2008 Offshore Technology Conference, Houston, Texas, U.S.A., 5–8 May 2008.
[8] Zhang, K., Wu, Y.S., Ding, C., Pruess, K., and Elmroth, E., *Parallel computing techniques for large-scale reservoir simulation of multi-component and multiphase fluid flow*, Paper SPE 66343, Proceedings of the 2001 SPE Reservoir Simulation Symposium, Houston, Texas, 2001.
[9] Wu, Y.S., Zhang, K., Ding, C., Pruess, K., Elmroth, E., and Bodvarsson, G.S., *An efficient parallel-computing scheme for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media, Advances In Water Resources*, 2002, 25, 243–261.
[10] Karypsis, G. and Kumar, V., *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, V4.0*, Technical Report, Department of Computer Science, University of Minnesota, 1998.
[11] Tuminaro, R.S., Heroux, M., Hutchinson, S.A. and Shadid, J.N., *Official Aztec User's Guide, Ver 2.1*, Massively Parallel Computing Research Laboratory, Sandia National Laboratories, Albuquerque, NM, 1999.
[12] Sloan, E.D., *Clathrate Hydrates of Nautral Gases*. Marcel Decker, Inc., New York, NY, 1998.
[13] Elmroth, E., Ding, C., and Wu, Y.S., *High performance computations for large-scale simulations of subsurface multiphase fluid and heat flow*, The Journal of Supercomputing, 2001, 18(3), 233–256.
[14] Zhang, K. and Wu, Y.S., *Enhancing scalability and efficiency of the TOUGH_MP for Linux clusters*, Proceedings of TOUGH Symposium 2006, Berkeley, CA, 2006.
[15] Smith, S., Boswell, R., Collett, T., Lee, M., Jones, E., Alaminos Canyon Block 818: *A documented example of gas hydrate saturated sand in the Gulf of Mexico, Fire in the Ice*, NETL Methane Hydrates R&D Program Newsletter, Fall 2006.
[16] Moridis, G.J., Reagan, M.T., and Zhang, K., *On the performance of Class 2 and Class 3 hydrate deposits during co-production with conventional gas*, OTC 19435, 2008 Offshore Technology Conference, Houston, Texas, U.S.A., 5–8 May, 2008
[17] Moridis, G.J., and Collett, T.S., *Strategies for gas production from hydrate accumulations under various geologic conditions*, Report LBNL-52568, Lawrence Berkeley National Laboratory, Berkeley, CA, 2003.